ON EFFICIENTLY COMPUTING THE EIGENVALUES OF LIMITED-MEMORY QUASI-NEWTON MATRICES

JENNIFER B. ERWAY* AND ROUMMEL F. MARCIA†

Abstract. In this paper, we consider the problem of efficiently computing the eigenvalues of limited-memory quasi-Newton matrices that exhibit a compact formulation. In addition, we produce a compact formula for quasi-Newton matrices generated by any member of the Broyden convex class of updates. Our proposed method makes use of efficient updates to the QR factorization that substantially reduces the cost of computing the eigenvalues after the quasi-Newton matrix is updated. Numerical experiments suggest that the proposed method is able to compute eigenvalues to high accuracy. Applications for this work include modified quasi-Newton methods and trust-region methods for large-scale optimization, the efficient computation of condition numbers and singular values, and sensitivity analysis.

Key words. Limited-memory quasi-Newton methods, quasi-Newton matrices, eigenvalues, spectral decomposition, QR decomposition

AMS subject classifications. 49M15, 15A12, 15A18, 65F15, 65K99, 90C06, 90C30, 90C53

1. Introduction.

Newton's method for minimizing a twice-continuously differentiable real-valued function $f : \Re^n \to \Re$ requires solving linear systems of the following form:

$$\nabla^2 f(x_k) p_k = -\nabla f(x_k). \tag{1.1}$$

When computing the Hessian is prohibitively expensive (e.g., n is very large) or $\nabla^2 f(x_k)$ is unavailable, solving (1.1) is impractical or impossible. In these cases, methods that only require first-order information can be used in place of a pure Newton method. Some of the most commonly-used first-order methods are quasi-Newton methods, which employ previous iterates and gradients to define a quasi-Newton approximation B_k to $\nabla^2 f(x_k)$, for each k. Conventional quasi-Newton matrices include Broyden-Fletcher-Goldfarb-Shanno (BFGS), Davidon-Fletcher-Powell (DFP), symmetric rank-one (SR1), and the Broyden convex class. In large-scale applications *limited-memory* versions of these quasi-Newton matrices are often used since they require less storage.

In this paper we demonstrate how to efficiently compute the eigenvalues of these limited-memory quasi-Newton matrices–allowing tools and methods previously restricted to small or medium-scale problems to be used in a large-scale quasi-Newton setting. For example, the ability to efficiently compute eigenvalues of these matrices allows for the use of traditional modified Newton methods (see, e.g., [17, 26]). In these methods, a *modified* Newton step is computed from solving a system of the form

$$\tilde{B}_k p_k = -\nabla f(x_k)$$

where $\tilde{B}_k$ is a positive-definite modification of B_k obtained by (i) replacing any negative eigenvalues with their absolute value in the spectral decomposition and (ii) thresholding to prevent small eigenvalues. (For more details on modified Newton methods, please see, e.g., [17, Section 11.4] or [26, Section 3.4].) Knowledge of the

*Department of Mathematics, PO Box 7388, Wake Forest University, Winston-Salem, NC 27109 (erwayjb@wfu.edu). Research supported in part by NSF grant CMMI-1334042

†School of Natural Sciences, University of California, Merced, 5200 N. Lake Road, Merced, CA 95343 (rmarcia@ucmerced.edu). Research supported in part by NSF grant CMMI-1333326

eigenvalues of a quasi-Newton matrix are also important in trust-region methods. In these methods, one of the challenges with using indefinite quasi-Newton matrices is in dealing with the so-called "hard case", which occurs when the trust-region subproblem does not have a solution on the boundary of the trust region and $\nabla f(x_k)$ is perpendicular to the eigenspace associated with the most negative eigenvalue [8, 13, 23]. However, knowing the leftmost eigenvalue provides an important bound in the "hard case" that simplifies computations considerably (see, e.g., [8, Algorithm 7.3.6]).

Knowing the spectrum of the BFGS, DFP, SR1 or any member of the Broyden convex class also enables the computation of their condition numbers and singular values [16]. From these, practical strategies may be devised to avoid matrices that are poorly scaled or nearly singular. Moreover, normwise sensitivity analysis for linear solves (e.g., forward and backward error bounds) often use the condition number, the two-norm, or the Frobenius norm of the system matrix (see e.g., [16, Chapter 2.7] or [18]); with the results of this paper, these bounds can be explicitly computed.

Our proposed method for efficiently computing eigenvalues of limited-memory quasi-Newton matrices relies on compact representations of these matrices. It is well-known that compact representations of BFGS, DFP, and SR1 matrices are available [5, 6, 11]; in particular, if B is generated using any of these updates with an initial Hessian approximation $B_0 = \gamma I$, $\gamma \in \Re$, then B can be written in the form

$$B = \gamma I + \Psi M \Psi^T, \tag{1.2}$$

where M is symmetric. With this compact representation in hand, the eigenvalue computation makes use of the QR factorization of Ψ. This method was first proposed by Burdakov et al. [4]. The bulk of the computational effort involves computing the QR factorization of Ψ.

This paper has two main contributions: (1) The compact representation of the Broyden convex class of updates, and (2) the efficient updating of the QR factorization used for the eigenvalue decomposition. We note that while compact representations of BFGS and DFP matrices are known [5, 6, 11], to our knowledge there has been no work done on generalizing the compact representation to the entire Broyden convex class of updates. This new compact representation allows us to extend the eigenvalue computation to any member of the Broyden convex class of updates.

Prior work on explicitly computing eigenvalues of quasi-Newton matrices has been restricted to at most two updates. A theorem by Wilkinson [27, pp. 94–97] can be used to compute the eigenvalues of a quasi-Newton matrix after one rank-one update. For more than one rank-one update, Wilkinson's theorem only provides bounds on eigenvalues. The eigenvalues of rank-one modifications are considered by Golub [15] and several methods are proposed, including Newton's method on the characteristic equation, linear interpolation on a related tridiagonal generalized eigenvalue problem, and finding zeros of the secular equation. Bunch et al. [3] extend the work of Golub to the case of eigenvalue algebraic multiplicity of more than one. Eigenvalue computations for more than one rank-one update are not proposed and, as is, these methods cannot be used to compute the eigenvalues for the general Broyden convex class of updates. Apostolopoulou et al. [2] and Apostolopoulou et al. [1] compute the eigenvalues of *minimal*-memory BFGS matrices, where the number of BFGS updates is limited to at most two. In these papers, formulas for the characteristic polynomials are derived that may be solved analytically. Due to the complexity involved in formulating characteristic polynomials and root finding, these approaches cannot

be generalized to handle more than two updates. (In Appendix A, we show how the same characteristic polynomial for the case of one update as in [1, 2] can be derived using our proposed approach.)

This paper is organized in six sections. In Section 2, we outline the compact formulations for the BFGS, DFP, and SR1 matrices. In Section 3, we present the compact formulation for the Broyden convex class of updates. The method to compute the eigenvalues of any limited-memory quasi-Newton matrix with the compact formulation (1.2) is given in Section 4. An efficient method to update the QR factorization of Ψ is also given in this section. In Section 5 we demonstrate the accuracy of the proposed method on a variety of limited-memory quasi-Newton matrices. Finally, in Section 6, there are some concluding remarks.

2. Compact formulations of quasi-Newton matrices. In this section, we review compact formulations of some of the most widely-used quasi-Newton matrices; in particular, we consider the BFGS, DFP, and SR1 matrices. First, we introduce notation and assumptions used throughout this paper.

Given a continuously differentiable function $f(x) \in \Re$ and iterates $\{x_k\}$, the quasi-Newton pairs $\{s_i, y_i\}$ are defined as follows:

$$s_i \triangleq x_{i+1} - x_i \quad \text{and} \quad y_i \triangleq \nabla f(x_{i+1}) - \nabla f(x_i),$$

where ∇f denotes the gradient of f.

The goal of this section is to express a quasi-Newton matrix obtained from these updates in the form

$$B_{k+1} \;=\; B_0 + \Psi_k M_k \Psi_k^T, \tag{2.1}$$

where $\Psi_k \in \Re^{n\times l}$, $M_k \in \Re^{l\times l}$, and B_0 is a diagonal matrix (i.e., $B_0 = \gamma I$, $\gamma \in \Re$). We will obtain factorizations of the form (2.1) where $l = k+1$ or $l = 2(k+1)$; in either case, we assume $l \ll n$.

Throughout this section, we make use of the following matrices:

$$S_k \triangleq [\; s_0 \;\; s_1 \;\; s_2 \;\; \cdots \;\; s_k \;] \;\in\; \Re^{n\times(k+1)},$$
$$Y_k \triangleq [\; y_0 \;\; y_1 \;\; y_2 \;\; \cdots \;\; y_k \;] \;\in\; \Re^{n\times(k+1)}.$$

Furthermore, we make use of the following decomposition of $S_k^T Y_k \in \Re^{(k+1)\times(k+1)}$:

$$S_k^T Y_k = L_k + D_k + R_k,$$

where L_k is strictly lower triangular, D_k is diagonal, and R_k is strictly upper triangular. We assume all updates are well-defined; for example, for the BFGS and DFP updates, we assume that $s_i^T y_i > 0$ for $i = 0, 1, \ldots, k$.

2.1. The BFGS update. The Broyden-Fletcher-Goldfarb-Shanno (BFGS) update is given by

$$B_{k+1} = B_k - \frac{1}{s_k^T B_k s_k} B_k s_k s_k^T B_k + \frac{1}{y_k^T s_k} y_k y_k^T,$$

where B_0 is a positive scalar multiple of the identity. The BFGS update is the most widely-used rank-two update formula that (i) satisfies the *quasi-Newton condition* $B_{k+1}s_k = y_k$, (ii) has hereditary symmetry, and (iii) provided that $y_i^T s_i > 0$ for $i = 0, \dots k$, then $\{B_k\}$ exhibits hereditary positive-definiteness. (For more background on the BFGS update formula, see, e.g., [17] or [26].)

We now consider compact formulations of the BFGS updates. Byrd et al. [6, Theorem 2.3] showed that B_{k+1} can be written in the form

$$B_{k+1} = B_0 + \Psi_k \Gamma_k^{-1} \Psi_k^T, \tag{2.2}$$

where

$$\Psi_k = \begin{pmatrix} B_0 S_k & Y_k \end{pmatrix} \quad \text{and} \quad \Gamma_k = -\begin{pmatrix} S_k^T B_0 S_k & L_k \\ L_k^T & -D_k \end{pmatrix}. \tag{2.3}$$

Defining $M_k \triangleq \Gamma_k^{-1}$ gives us the desired compact form (2.1) with $l = 2(k+1)$.

2.2. The DFP update. The Davidon-Fletcher-Powell (DFP) update is derived from applying BFGS updates to approximate the inverse of the Hessian. The DFP update formula is given by

$$B_{k+1} = \left(I - \frac{y_k s_k^T}{y_k^T s_k}\right) B_k \left(I - \frac{s_k y_k^T}{y_k^T s_k}\right) + \frac{y_k y_k^T}{y_k^T s_k}$$

Like BFGS, DFP updates satisfy the quasi-Newton condition while exhibiting hereditary symmetry and positive-definiteness. (For more background on the DFP update formula, see, e.g., [17] or [26].)

The compact formulation for the DFP update is found in [11, Theorem 1], where Erway et al. showed that B_{k+1} can be written in the form (2.1) with

$$\Psi_k = \begin{pmatrix} B_0 S_k & Y_k \end{pmatrix} \quad \text{and} \quad M_k = \begin{pmatrix} 0 & -\bar{L}_k^{-T} \\ -\bar{L}_k^{-1} & \bar{L}_k^{-1}(D_k + S_k^T B_0 S_k)\bar{L}_k^{-T} \end{pmatrix}, \tag{2.4}$$

where $\bar{L}_k \triangleq L_k + D_k$. In this case, $l = 2(k+1)$. We note that in [11], $\Psi_k M_k \Psi_k^T$ is expressed as the equivalent product

$$\Psi_k M_k \Psi_k^T = \begin{pmatrix} Y_k & B_0 S_k \end{pmatrix} \begin{pmatrix} \bar{L}_k^{-1}(D_k + S_k^T B_0 S_k)\bar{L}_k^{-T} & -\bar{L}_k^{-1} \\ -\bar{L}_k^{-T} & 0 \end{pmatrix} \begin{pmatrix} Y_k^T \\ (B_0 S_k)^T \end{pmatrix}.$$

2.3. The SR1 update. The symmetric rank-one (SR1) update formula is given by

$$B_{k+1} = B_k + \frac{1}{s_k^T (y_k - B_k s_k)} (y_k - B_k s_k)(y_k - B_k s_k)^T. \tag{2.5}$$

The SR1 update is the unique rank-one update that satisfies the quasi-Newton condition $B_{k+1}s_k = y_k$ and exhibits hereditary symmetry. Unlike BFGS and DFP, these matrices do not exhibit hereditary positive-definiteness. In fact, even if $y_i^T s_i > 0$ for each $i = 0, \dots, k$, the sequence $\{B_k\}$ may not be positive definite. The SR1 update tends to be a better approximation of the true Hessian since it is allowed to take on negative curvature; moreover, it has known convergence properties superior to other widely-used quasi-Newton methods such as BFGS [7]. However, when using these updates extra precaution must be taken so the denominator $s_k^T(y_k - B_k s_k)$ is nonzero. (For more background on the SR1 update formula, please see, e.g., [26] or [17].)

The compact formulation for the SR1 update is due to Byrd et al. [6, Theorem 5.1], where they showed that B_{k+1} can be written in the form (2.1) with

$$\Psi_k \;=\; Y_k - B_0 S_k \quad \text{and} \quad M_k \;=\; (D_k + L_k + L_k^T - S_k^T B_0 S_k)^{-1}.$$

Note that in the SR1 case, $l = k + 1$.

3. The Broyden convex class of updates. In this section, we present a compact formulation for the Broyden convex class of updates. The Broyden convex class of updates is given by

$$B_{k+1}^{\phi} = B_k^{\phi} - \frac{1}{s_k^T B_k^{\phi} s_k} B_k^{\phi} s_k s_k^T B_k^{\phi} + \frac{1}{y_k^T s_k} y_k y_k^T + \phi (s_k^T B_k^{\phi} s_k) w_k w_k^T, \tag{3.1}$$

where $\phi \in [0, 1]$ and

$$w_k = \frac{y_k}{y_k^T s_k} - \frac{B_k^{\phi} s_k}{s_k^T B_k^{\phi} s_k},$$

(see, e.g., [22, 17]). Both the BFGS and the DFP updates are members of this family. (Setting $\phi = 0$ gives the BFGS update, and setting $\phi = 1$ yields the DFP update.) In fact, this class of updates can be expressed in terms of the BFGS and DFP updates:

$$B_{k+1}^{\phi} = (1 - \phi) B_{k+1}^{BFGS} + \phi B_{k+1}^{DFP}, \tag{3.2}$$

where $\phi \in [0, 1]$ and

$$B_{k+1}^{BFGS} = B_k^{\phi} - \frac{1}{s_k^T B_k^{\phi} s_k} B_k^{\phi} s_k s_k^T B_k^{\phi} + \frac{1}{y_k^T s_k} y_k y_k^T,$$

and

$$B_{k+1}^{DFP} = \left(I - \frac{y_k s_k^T}{y_k^T s_k} \right) B_k^{\phi} \left(I - \frac{s_k y_k^T}{y_k^T s_k} \right) + \frac{y_k y_k^T}{y_k^T s_k}.$$

(In other words, B_{k+1}^{ϕ} is the Broyden convex class matrix B_k^{ϕ} updated using the BFGS update and the DFP update, respectively.) All updates in this class satisfy the quasi-Newton condition. Moreover, members of this class enjoy hereditary symmetry and positive-definiteness provided $y_i^T s_i > 0$ for all i. Dixon [10] shows that, under some conditions, the iterates generated using a quasi-Newton method belonging to the Broyden class of convex updates together with an exact line search will be identical in direction; in fact, the choice of ϕ only affects the step length of the search direction (see [12] for a detailed explanation). In practice, for general nonlinear functions, exact line searches are impractical; with inexact line searches, it is well known that members of the Broyden convex class of updates can behave significantly differently. (For more background and analysis of the Broyden convex class of updates, see [12].)

We now consider compact formulations of the Broyden convex class of updates. For notational simplicity, we drop the superscript ϕ for the duration of this paper. To find the compact formulation, we expand (3.1) to obtain

$$\begin{aligned} B_{k+1} = B_k &- \frac{1 - \phi}{s_k^T B_k s_k} B_k s_k s_k^T B_k - \frac{\phi}{y_k^T s_k} B_k s_k y_k^T - \frac{\phi}{y_k^T s_k} y_k s_k^T B_k \\ &+ \left(1 + \phi \frac{s_k^T B_k s_k}{y_k^T s_k} \right) \frac{1}{y_k^T s_k} y_k y_k^T, \end{aligned}$$

and thus, B_{k+1} can be written compactly as

$$B_{k+1} = B_k + \begin{pmatrix} B_k s_k & y_k \end{pmatrix} \begin{pmatrix} -\dfrac{(1-\phi)}{s_k^T B_k s_k} & -\dfrac{\phi}{y_k^T s_k} \\ -\dfrac{\phi}{y_k^T s_k} & \left(1 + \phi \dfrac{s_k^T B_k s_k}{y_k^T s_k}\right) \dfrac{1}{y_k^T s_k} \end{pmatrix} \begin{pmatrix} (B_k s_k)^T \\ y_k^T \end{pmatrix}. \tag{3.3}$$

Recall that our goal is to write B_{k+1} in the form

$$B_{k+1} = B_0 + \Psi_k M_k \Psi_k^T,$$

where $\Psi_k \in \Re^{n \times 2(k+1)}$ and $M_k \in \Re^{2(k+1) \times 2(k+1)}$. Letting Ψ_k be defined as

$$\Psi_k \triangleq \begin{pmatrix} B_0 s_0 & B_0 s_1 & \cdots & B_0 s_k & y_0 & y_1 & \cdots & y_k \end{pmatrix} = \begin{pmatrix} B_0 S_k & Y_k \end{pmatrix}, \tag{3.4}$$

we now derive an expression for M_k.

3.1. General M_k. In this section we state and prove a theorem that gives an expression for M_k. The eigenvalue computation in Section 5 requires the ability to form M_k. For this reason, we also provide a practical recursive method for computing M_k.

Theorem 1. *Let $\Lambda_k \in \Re^{(k+1)\times(k+1)}$ be a diagonal matrix such that*

$$\Lambda_k = \underset{0 \le i \le k}{diag}\left(\lambda_i\right), \qquad \text{where } \lambda_i = \frac{1}{-\dfrac{1-\phi}{s_i^T B_i s_i} - \dfrac{\phi}{s_i^T y_i}} \text{ for } 0 \le i \le k. \tag{3.5}$$

If B_{k+1} is updated using the Broyden convex class of updates (3.1), where $\phi \in [0,1]$, then B_{k+1} can be written as $B_{k+1} = B_0 + \Psi_k M_k \Psi_k^T$, where Ψ_k is defined as in (3.4) and

$$M_k = \begin{pmatrix} -S_k^T B_0 S_k + \phi\Lambda_k & -L_k + \phi\Lambda_k \\ -L_k^T + \phi\Lambda_k & D_k + \phi\Lambda_k \end{pmatrix}^{-1}. \tag{3.6}$$

Proof. This proof is broken into two parts. First, we consider the special cases when $\phi = 0$ and $\phi = 1$. Then, we prove by induction the case when $\phi \in (0,1)$.

When $\phi = 0$, (3.1) becomes the BFGS update and M_k in (3.6) simplifies to

$$M_k = \begin{pmatrix} -S_k^T B_0 S_k & -L_k \\ -L_k^T & D_k \end{pmatrix}^{-1},$$

which is consistent with (2.2) and (2.3). When $\phi = 1$, then $\Lambda_k = -D_k$ and so (3.1) is the DFP update and with

$$M_k = \begin{pmatrix} -S_k^T B_0 S_k - D_k & -\bar{L}_k \\ -\bar{L}_k^T & 0 \end{pmatrix}^{-1},$$

where $\bar{L} = L_k + D_k$. After some algebra, it can be shown that this is exactly M_k given in (2.4). Thus, M_k in (3.6) is correct for $\phi = 0$ and $\phi = 1$.

The proof for $\phi \in (0,1)$ is by induction on k. We begin by considering the base case $k = 0$. For $k = 0$, B_1 is given by (3.3), and thus, $B_1 = B_0 + \Psi_0 \widehat{M}_0 \Psi_0^T$ where $\Psi_0 = (\ B_0 s_0 \ \ y_0\ )$ and

$$\widehat{M}_0 \triangleq \begin{pmatrix} -\dfrac{(1-\phi)}{s_0^T B_0 s_0} & -\dfrac{\phi}{y_0^T s_0} \\ -\dfrac{\phi}{y_0^T s_0} & \left(1 + \phi \dfrac{s_0^T B_0 s_0}{y_0^T s_0}\right) \dfrac{1}{y_0^T s_0} \end{pmatrix}. \tag{3.7}$$

To complete the base case, we now show that $\widehat{M}_0$ in (3.7) is equivalent to M_0 in (3.6). For simplicity, $\widehat{M}_0$ can be written as

$$\widehat{M}_0 = \begin{pmatrix} \alpha_0 & \beta_0 \\ \beta_0 & \delta_0 \end{pmatrix}, \tag{3.8}$$

where

$$\alpha_0 = -\frac{(1-\phi)}{s_0^T B_0 s_0}, \quad \beta_0 = -\frac{\phi}{y_0^T s_0}, \quad \text{and} \quad \delta_0 = \left(1 + \phi \frac{s_0^T B_0 s_0}{y_0^T s_0}\right) \frac{1}{y_0^T s_0}. \tag{3.9}$$

We note that α_0 and β_0 are nonzero since $0 < \phi < 1$. Consequently, δ_0 can be written as

$$\delta_0 = \left(1 + \phi \frac{s_0^T B_0 s_0}{y_0^T s_0}\right) \frac{1}{y_0^T s_0} = -\left(1 + (1-\phi)\frac{\beta_0}{\alpha_0}\right) \frac{\beta_0}{\phi} = -\frac{\beta_0}{\phi} - \frac{\beta_0^2}{\phi \alpha_0} + \frac{\beta_0^2}{\alpha_0}. \tag{3.10}$$

The determinant, η_0, of $\widehat{M}_0$ can be written as

$$\eta_0 = \alpha_0 \delta_0 - \beta_0^2 \ = \ -\frac{\alpha_0 \beta_0}{\phi} - \frac{\beta_0^2}{\phi} \ = \ -\frac{\beta_0}{\phi}(\alpha_0 + \beta_0). \tag{3.11}$$

Since all members of the convex class are positive definite, both α_0 and β_0 are negative, and thus, $\alpha_0 + \beta_0 \neq 0$ and $\eta_0 \neq 0$ in (3.11). It follows that $\widehat{M}_0$ is invertible, and in particular,

$$\widehat{M}_0^{-1} = \begin{pmatrix} \delta_0/\eta_0 & -\beta_0/\eta_0 \\ -\beta_0/\eta_0 & \alpha_0/\eta_0 \end{pmatrix}.$$

Together with (3.10), the (1,1) entry of $\widehat{M}_0^{-1}$ simplifies to

$$\begin{aligned} \frac{\delta_0}{\eta_0} &= \frac{-\left(\dfrac{\alpha_0 + \beta_0}{\alpha_0} - \dfrac{\phi\beta_0}{\alpha_0}\right)\dfrac{\beta_0}{\phi}}{-\dfrac{\beta_0}{\phi}(\alpha_0 + \beta_0)} \\ &= \frac{1}{\alpha_0} - \frac{\phi\beta_0}{\alpha_0(\alpha_0 + \beta_0)} \\ &= \frac{(\alpha_0 + \beta_0)(1-\phi) + \phi\alpha_0}{\alpha_0(\alpha_0 + \beta_0)} \\ &= \frac{1-\phi}{\alpha_0} + \frac{\phi}{\alpha_0 + \beta_0}. \end{aligned} \tag{3.12}$$

Finally, the (2,2) entry of $\widehat{M}_0^{-1}$ can be written as

$$\frac{\alpha_0}{\eta_0} = -\frac{\phi\alpha_0}{\beta_0(\alpha_0+\beta_0)} \;=\; -\frac{\phi}{\beta_0} + \frac{\phi}{\alpha_0+\beta_0}. \tag{3.13}$$

Thus, combining (3.11), (3.12), and (3.13), we obtain the following equivalent expression for $\widehat{M}_0^{-1}$:

$$\widehat{M}_0^{-1} = \begin{pmatrix} \dfrac{1-\phi}{\alpha_0} + \dfrac{\phi}{\alpha_0+\beta_0} & \dfrac{\phi}{\alpha_0+\beta_0} \\ \dfrac{\phi}{\alpha_0+\beta_0} & -\dfrac{\phi}{\beta_0} + \dfrac{\phi}{\alpha_0+\beta_0} \end{pmatrix}. \tag{3.14}$$

For the case $k = 0$, $L_0 = R_0 = 0$; moreover, $\lambda_0 = 1/(\alpha_0+\beta_0)$. Substituting back in for α_0, β_0 and δ_0 using (3.9), we obtain

$$\begin{aligned} \widehat{M}_0 &= \begin{pmatrix} -s_0^T B_0 s_0 + \phi\lambda_0 & \phi\lambda_0 \\ \phi\lambda_0 & s_0^T y_0 + \phi\lambda_0 \end{pmatrix}^{-1}, \\ &= \begin{pmatrix} -S_k^T B_0 S_k + \phi\Lambda_k & -L_k + \phi\Lambda_k \\ -L_k^T + \phi\Lambda_k & D_k + \phi\Lambda_k \end{pmatrix}^{-1} \\ &= M_0, \end{aligned}$$

proving the base case.

For the induction step, assume

$$B_m = B_0 + \Psi_{m-1} M_{m-1} \Psi_{m-1}^T, \tag{3.15}$$

where Ψ_{m-1} is defined as in (3.4) and

$$M_{m-1} = \begin{pmatrix} -S_{m-1}^T B_0 S_{m-1} + \phi\Lambda_{m-1} & -L_{m-1} + \phi\Lambda_{m-1} \\ -L_{m-1}^T + \phi\Lambda_{m-1} & D_{m-1} + \phi\Lambda_{m-1} \end{pmatrix}^{-1}. \tag{3.16}$$

From (3.3), we have

$$B_{m+1} = B_0 + \Psi_{m-1} M_{m-1} \Psi_{m-1}^T + \begin{pmatrix} B_m s_m & y_m \end{pmatrix} \begin{pmatrix} \alpha_m & \beta_m \\ \beta_m & \delta_m \end{pmatrix} \begin{pmatrix} (B_m s_m)^T \\ y_m^T \end{pmatrix}, \tag{3.17}$$

where

$$\alpha_m = -\frac{1-\phi}{s_m^T B_m s_m}, \quad \beta_m = -\frac{\phi}{y_m^T s_m},$$

and

$$\delta_m = \left(1 + \phi\frac{s_m^T B_m s_m}{y_m^T s_m}\right)\frac{1}{y_m^T s_m} = -\left(1 + (1-\phi)\frac{\beta_m}{\alpha_m}\right)\frac{\beta_m}{\phi}.$$

As in the base case, $k = 0$, we note that α_m and β_m are nonzero since $0 < \phi < 1$, and that the determinant $\alpha_m\delta_m - \beta_m^2$ is also nonzero.

Multiplying (3.15) by s_m on the right, we obtain

$$B_m s_m = B_0 s_m + \Psi_{m-1} M_{m-1} \Psi_{m-1}^T s_m. \tag{3.18}$$

Then, substituting this into (3.17) yields

(3.19)
$$B_{m+1} = B_0 + \Psi_{m-1} M_{m-1} \Psi_{m-1}^T + \begin{pmatrix} B_0 s_m + \Psi_{m-1} p_m & y_m \end{pmatrix} \begin{pmatrix} \alpha_m & \beta_m \\ \beta_m & \delta_m \end{pmatrix} \begin{pmatrix} (B_0 s_m + \Psi_{m-1} p_m)^T \\ y_m^T \end{pmatrix},$$

where $p_m \triangleq M_{m-1} \Psi_{m-1}^T s_m$. Equivalently,

(3.20)
$$B_{m+1} = B_0 + \begin{pmatrix} \Psi_{m-1} & B_0 s_m & y_m \end{pmatrix} \begin{pmatrix} M_{m-1} + \alpha_m p_m p_m^T & \alpha_m p_m & \beta_m p_m \\ \alpha_m p_m^T & \alpha_m & \beta_m \\ \beta_m p_m^T & \beta_m & \delta_m \end{pmatrix} \begin{pmatrix} \Psi_{m-1}^T \\ (B_0 s_m)^T \\ y_m^T \end{pmatrix}.$$

The 3×3 block matrix in (3.20) has the following decomposition:

(3.21)
$$\begin{pmatrix} M_{m-1} + \alpha_m p_m p_m^T & \alpha_m p_m & \beta_m p_m \\ \alpha_m p_m^T & \alpha_m & \beta_m \\ \beta_m p_m^T & \beta_m & \delta_m \end{pmatrix} = \begin{pmatrix} I & p_m & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} M_{m-1} & 0 & 0 \\ 0 & \alpha_m & \beta_m \\ 0 & \beta_m & \delta_m \end{pmatrix} \begin{pmatrix} I & 0 & 0 \\ p_m^T & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

allowing us to compute its inverse as follows:

$$\begin{aligned} \begin{pmatrix} M_{m-1} + \alpha_m p_m p_m^T & \alpha_m p_m & \beta_m p_m \\ \alpha_m p_m^T & \alpha_m & \beta_m \\ \beta_m p_m^T & \beta_m & \delta_m \end{pmatrix}^{-1} &= \begin{pmatrix} I & 0 & 0 \\ -p_m^T & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} M_{m-1}^{-1} & 0 & 0 \\ 0 & \tilde{\alpha}_m & \tilde{\beta}_m \\ 0 & \tilde{\beta}_m & \tilde{\delta}_m \end{pmatrix} \begin{pmatrix} I & -p_m & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} M_{m-1}^{-1} & -M_{m-1}^{-1} p_m & 0 \\ -p_m^T M_{m-1}^{-1} & p_m^T M_{m-1}^{-1} p_m + \tilde{\alpha}_m & \tilde{\beta}_m \\ 0 & \tilde{\beta}_m & \tilde{\delta}_m \end{pmatrix}, \end{aligned} \tag{3.22}$$

where

$$\begin{aligned} \tilde{\alpha}_m &= \frac{\delta_m}{\alpha_m \delta_m - \beta_m^2} = \frac{1-\phi}{\alpha_m} + \frac{\phi}{\alpha_m + \beta_m} \\ \tilde{\beta}_m &= \frac{-\beta_m}{\alpha_m \delta_m - \beta_m^2} = \frac{\phi}{\alpha_m + \beta_m} \\ \tilde{\delta}_m &= \frac{\alpha_m}{\alpha_m \delta_m - \beta_m^2} = -\frac{\phi}{\beta_0} + \frac{\phi}{\alpha_0 + \beta_0}. \end{aligned}$$

We now simplify the entries of (3.22). Since $p_m = M_{m-1} \Psi_{m-1}^T s_m$, then $M_{m-1}^{-1} p_m = \Psi_{m-1}^T s_m$, giving us an expression for the (1,2) and (2,1) entries. The (2,2) block entry is simplified by first multiplying (3.18) by s_m^T on the left to obtain $s_m^T B_m s_m = s_m^T B_0 s_m + p_m^T M_{m-1}^{-1} p_m$. Then,

$$\begin{aligned} p_m^T M_{m-1}^{-1} p_m + \tilde{\alpha}_m &= -s_m^T B_0 s_m + s_m^T B_m s_m + \tilde{\alpha}_m \\ &= -s_m^T B_0 s_m - \frac{1-\phi}{\alpha_m} + \frac{1-\phi}{\alpha_m} + \frac{\phi}{\alpha_m + \beta_m} \\ &= -s_m^T B_0 s_m + \frac{\phi}{\alpha_m + \beta_m}. \end{aligned}$$

Thus, (3.22) can be written as

$$\begin{pmatrix} M_{m-1}^{-1} & -\Psi_{m-1}^T s_m & 0 \\ -s_m^T \Psi_{m-1} & -s_m^T B_0 s_m + \dfrac{\phi}{\alpha_m + \beta_m} & \dfrac{\phi}{\alpha_m + \beta_m} \\ 0 & \dfrac{\phi}{\alpha_m + \beta_m} & y_m^T s_m + \dfrac{\phi}{\alpha_m + \beta_m} \end{pmatrix}. \tag{3.23}$$

We now show (3.6) holds using (3.23). Define the permutation matrix $\Pi_m \in \Re^{2(m+1)\times 2(m+1)}$ as follows:

$$\Pi_m = \begin{pmatrix} I_m & & & \\ & 0 & I_m & \\ & 1 & 0 & \\ & & & 1 \end{pmatrix}. \tag{3.24}$$

Then,

$$[\Psi_{m-1} \;\; B_0 s_m \;\; y_m]\,\Pi_m = \Psi_m;$$

in other words, $[\Psi_{m-1} \;\; B_0 s_m \;\; y_m] = \Psi_m \Pi_m^T$. Therefore, (3.20) can be written as

$$B_{m+1} = B_0 + \Psi_m \Pi_m^T \widehat{M}_m \Pi_m \Psi_m^T,$$

where

$$\widehat{M}_m \triangleq \begin{pmatrix} M_{m-1} + \alpha_m p_m p_m^T & \alpha_m p_m & \beta_m p_m \\ \alpha_m p_m^T & \alpha_m & \beta_m \\ \beta_m p_m^T & \beta_m & \delta_m \end{pmatrix}.$$

It remains to show (3.6) holds for $k = m$ with $M_m \triangleq \Pi_m^T \widehat{M}_m \Pi_m$.

We now consider M_m^{-1} given by

$$M_m^{-1} = \left(\Pi_m^T \begin{pmatrix} M_{m-1} + \alpha_m p_m p_m^T & \alpha_m p_m & \beta_m p_m \\ \alpha_m p_m^T & \alpha_m & \beta_m \\ \beta_m p_m^T & \beta_m & \delta_m \end{pmatrix} \Pi_m \right)^{-1}, \tag{3.25}$$

which can be simplified using (3.23):

$$M_m^{-1} = \Pi_m^T \begin{pmatrix} M_{m-1}^{-1} & -\Psi_{m-1}^T s_m & 0 \\ -s_m^T \Psi_{m-1} & -s_m^T B_0 s_m + \dfrac{\phi}{\alpha_m + \beta_m} & \dfrac{\phi}{\alpha_m + \beta_m} \\ 0 & \dfrac{\phi}{\alpha_m + \beta_m} & y_m^T s_m + \dfrac{\phi}{\alpha_m + \beta_m} \end{pmatrix} \Pi_m. \tag{3.26}$$

Now partition M_{m-1}^{-1} as follows:

$$M_{m-1}^{-1} \;=\; \begin{pmatrix} (M_{m-1}^{-1})_{11} & (M_{m-1}^{-1})_{12} \\ (M_{m-1}^{-1})_{21} & (M_{m-1}^{-1})_{22} \end{pmatrix}.$$

Applying the permutation matrices together with $\Psi_{m-1}^T s_m = \begin{pmatrix} S_{m-1}^T B_0 s_m \\ Y_{m-1}^T s_m \end{pmatrix}$, we have that

$$M_m^{-1} = \left(\begin{array}{cc|cc} (M_{m-1}^{-1})_{11} & -S_{m-1}^T B_0 s_m & (M_{m-1}^{-1})_{12} & 0 \\ -s_m^T B_0 S_{m-1} & -s_m^T B_0 s_m + \dfrac{\phi}{\alpha_m + \beta_m} & -s_m^T Y_m & \dfrac{\phi}{\alpha_m + \beta_m} \\ \hline (M_{m-1}^{-1})_{21} & -Y_m^T s_m & (M_{m-1}^{-1})_{22} & 0 \\ 0 & \dfrac{\phi}{\alpha_m + \beta_m} & 0 & y_m^T s_m + \dfrac{\phi}{\alpha_m + \beta_m} \end{array} \right).$$

Simplifying using the induction hypothesis (3.16) yields

$$M_m^{-1} = \left(\begin{array}{cc|cc} -S_{m-1}^T B_0 S_{m-1} + \phi\Lambda_{m-1} & -S_{m-1}^T B_0 s_m & -L_{m-1} + \phi\Lambda_{m-1} & 0 \\ -s_m^T B_0 S_{m-1} & -s_m^T B_0 s_m + \dfrac{\phi}{\alpha_m + \beta_m} & -s_m^T Y_m & \dfrac{\phi}{\alpha_m + \beta_m} \\ \hline -L_{k-1}^T + \phi\Lambda_{k-1} & -Y_m^T s_m & D_{k-1} + \phi\Lambda_{k-1} & 0 \\ 0 & \dfrac{\phi}{\alpha_m + \beta_m} & 0 & y_m^T s_m + \dfrac{\phi}{\alpha_m + \beta_m} \end{array}\right)$$
$$= \begin{pmatrix} -S_m^T B_0 S_m + \phi\Lambda_m & -L_m + \phi\Lambda_m \\ -L_m^T + \phi\Lambda_m & D_m + \phi\Lambda_m \end{pmatrix},$$

i.e., (3.6) holds for $k = m$. □

Although we have found an expression for M_k, computing M_k is not straightforward. In particular, the diagonal matrix Λ_k in Eq. (3.5) involves $s_i^T B_i s_i$, which requires B_i for $0 \le i \le k$. In the following section we propose a different way of computing M_k that does not necessitate storing the quasi-Newton matrices B_i for $0 \le i \le k$.

3.2. Computing M_k. In this section we propose a recursive method for computing M_k from M_{k-1}. We have shown that

$$M_k \;=\; \Pi_k^T \begin{pmatrix} M_{k-1} + \alpha_k p_k p_k^T & \alpha_k p_k & \beta_k p_k \\ \alpha_k p_k^T & \alpha_k & \beta_k \\ \beta_k p_k^T & \beta_k & \delta_k \end{pmatrix} \Pi_k.$$

The vector p_k can be computed as follows:

$$p_k = M_{k-1}\Psi_{k-1}^T s_k = M_{k-1}\begin{pmatrix} (B_0 S_{k-1})^T \\ Y_{k-1}^T \end{pmatrix} s_k = M_{k-1}\begin{pmatrix} S_{k-1}^T B_0 s_k \\ Y_{k-1}^T s_k \end{pmatrix}.$$

Note that $(S_{k-1}^T B_0 s_k)^T$ is the last row (save the diagonal entry) of $S_k^T B_0 S_k$ and $(Y_{k-1}^T s_k)^T$ is the last row (save the diagonal entry) of $S_k^T Y_k$. The entry α_k, which is given by $\alpha_k = -(1-\phi)/s_k^T B_k s_k$ can be computed from the following:

$$s_k^T B_k s_k = s_k^T \Big(B_0 + \Psi_{k-1} M_{k-1} \Psi_{k-1}^T \Big) s_k = s_k^T B_0 s_k + s_k^T \Psi_{k-1} p_k. \tag{3.27}$$

The quantity $s_k^T B_0 s_k$ is the last diagonal entry in $S_k^T B_0 S_k$, and $s_k^T \Psi_{k-1} p_k$ is the inner product of $\Psi_{k-1}^T s_k$ (which was formed when computing p_k) and p_k. The entry β_k is given by $\beta_k = -\phi/y_k^T s_k$, where $y_k^T s_k$ is the last diagonal entry in $S_k^T Y_k$. Finally, $\delta_k = (1 + \phi s_k^T B_k s_k / y_k^T s_k)/y_k^T s_k$, which uses the previously computed quantities $s_k^T B_k s_k$ and $y_k^T s_k$.

We summarize this recursive method in Algorithm 1.

Algorithm 1. This algorithm computes M_k in (3.6).

Define ϕ and B_0;
Define M_0 using (3.7);
Define $\Psi_0 = (B_0 s_0 \; y_0)$;

for $j = 1 : k$
$p_j \leftarrow M_{j-1}(\Psi_{j-1}^T s_j)$;
$s_j^T B_j s_j \leftarrow s_j^T B_0 s_j + (s_j^T \Psi_{j-1}) p_j$;
$\alpha_j \leftarrow -(1-\phi)/(s_j^T B_j s_j)$;
$\beta_j \leftarrow -\phi/(y_j^T s_j)$;
$\delta_j \leftarrow (1 + \phi(s_j^T B_j s_j)/(y_j^T s_j))/(y_j^T s_j)$;
$M_j \leftarrow \Pi_j^T \begin{pmatrix} M_{j-1} + \alpha_j p_j p_j^T & \alpha_j p_j & \beta_j p_j \\ \alpha_j p_j^T & \alpha_j & \beta_j \\ \beta_j p_j^T & \beta_j & \delta_j \end{pmatrix} \Pi_j$, where Π_j is as in (3.24);
end

The matrices $\{\Pi_j\}$ are not formed explicitly since they are permutation matrices; thus, no matrix-matrix products are required by the recursion algorithm.

4. Computing the eigenvalues of B_{k+1}. In this section, we demonstrate how to compute the eigenvalues of a limited-memory matrix B_{k+1} when the following decomposition is available:

$$B_{k+1} = B_0 + \Psi_k M_k \Psi_k^T, \tag{4.1}$$

where $B_0 = \gamma I$, $\gamma \in \Re$. We assume that $B_{k+1} \in \Re^{n\times n}$ but only m limited-memory updates are stored, where $m \ll n$ (see, e.g., [19, 21, 24, 25]). In large-scale optimization, typically $m < 10$ (e.g., Byrd et al. [6] recommend $m \in [2, 6]$).

The material presented in Section 4.1 was first proposed by Burdakov et al. [4] in a different manner. We explain these differences at the end of the section.

4.1. Eigenvalues via the QR decomposition. We begin by finding the eigenvalues of B_{k+1} when B_{k+1} is obtained using the Broyden convex class of updates; at the end of this section, we describe the modifications needed to find the eigenvalues for the SR1 case. We assume $k + 1 \le m \ll n$.

For the Broyden convex class of updates, $\Psi_k = \begin{pmatrix} B_0 S_k & Y_k \end{pmatrix}$, i.e.,

$$\Psi_k = \begin{pmatrix} B_0 s_0 & B_0 s_1 & \cdots & B_0 s_k & y_0 & y_1 & \cdots & y_k \end{pmatrix}.$$

To facilitate updating Ψ_k after computing a new limited-memory pair (see Section 4.2), we permute the columns of Ψ_k using a permutation matrix P (also called the "perfect shuffle"–see, e.g., [20]) so that

$$\hat{\Psi}_k \triangleq \Psi_k P = \begin{pmatrix} B_0 s_0 & y_0 & B_0 s_1 & y_1 & \cdots & B_0 s_k & y_k \end{pmatrix}.$$

Let

$$\hat{\Psi}_k = QR \in \Re^{n\times l}$$

be the QR decomposition of $\hat{\Psi}_k$, where $Q \in \Re^{n\times n}$ has orthonormal columns and $R \in \Re^{n\times l}$ is upper triangular (see, e.g., [16]).

Then,

$$\begin{aligned} B_{k+1} &= B_0 + \Psi_k M_k \Psi_k^T \\ &= B_0 + \hat{\Psi}_k P^T M_k P \hat{\Psi}_k^T \\ &= B_0 + QRP^T M_k P R^T Q^T \end{aligned}$$

The matrix $RP^T M_k P R^T$ is a real symmetric $n \times n$ matrix. Since $\hat{\Psi}_k \in \Re^{n \times l}$, R has at most rank l; moreover, R can be written in the form

$$R = \begin{pmatrix} R_1 \\ 0 \end{pmatrix},$$

where $R_1 \in \Re^{l \times l}$. Then,

$$RP^T M_k P R^T = \begin{pmatrix} R_1 \\ 0 \end{pmatrix} P^T M_k P \begin{pmatrix} R_1^T & 0 \end{pmatrix} = \begin{pmatrix} R_1 P^T M_k P R_1^T & 0 \\ 0 & 0 \end{pmatrix}.$$

The eigenvalues of $RP^T M_k P R^T$ can be explicitly computed by forming the spectral decomposition of $R_1 P^T M_k P R_1^T \in \Re^{l \times l}$. That is, suppose $V_1 D_1 V_1$ is the spectral decomposition of $R_1 P^T M_k P R_1^T$. Then,

$$RP^T M_k P R^T = \begin{pmatrix} R_1 P^T M_k P R_1^T & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} V_1 D_1 V_1^T & 0 \\ 0 & 0 \end{pmatrix} = V D V^T$$

where

$$V \triangleq \begin{pmatrix} V_1 & 0 \\ 0 & I \end{pmatrix} \in \Re^{n \times n} \qquad \text{and} \qquad D \triangleq \begin{pmatrix} D_1 & 0 \\ 0 & 0 \end{pmatrix} \in \Re^{n \times n}.$$

This gives that

$$\begin{aligned} B_{k+1} &= B_0 + QVDV^TQ^T \\ &= \gamma I + QVDV^TQ^T \\ &= QV(\gamma I + D)V^TQ^T, \end{aligned} \tag{4.2}$$

yielding the spectral decomposition of B_{k+1}. The matrix B_{k+1} has an eigenvalue of γ with multiplicity $n - l$ and l eigenvalues given by $\gamma + d_i$, where $1 \leq i \leq l$. In practice, the matrices Q and V in (4.2) are not stored.

In the case of the SR1 updates, $\Psi_k = Y_k - B_0 S_k$ and no permutation matrix is required (i.e., $P = I$).

Computing the eigenvalues of B_{k+1} directly is an $O(n^3)$ process. In contrast, the above decomposition requires the QR factorization of Ψ_k and the eigendecomposition of $R_1 P^T M_k P R_1^T$, requiring ($O(nl^2)$ flops) and $O(l^3)$ flops, respectively. Since $l \ll n$, the proposed method results in substantial computational savings.

The material presented above was first presented in [4] using the so-called "thin QR" factorization together with a Choleksy factorization of an $m \times m$ symmetric matrix. We chose to present the eigenvalue decomposition in terms of "full QR" since we are able update the QR factorization as new quasi-Newton pairs are computed without having to store Q explicitly. Here we describe in detail how the update to the QR factorization can be performed–enabling the eigenvalues of the updated quasi-Newton matrix to be computed efficiently.

4.2. Handling updates to $\hat{\Psi}$. In this section we detail handling updates to the QR decomposition of $\hat{\Psi}_k$ when additional limited-memory pairs are added to S and Y. We consider two cases: Adding a limited-memory pair (s_{k+1}, y_{k+1}) when $k + 1 < m$ and when $k + 1 \geq m$, where m is the maximum number of limited-memory updates allowed to be stored. The case $k + 1 < m$ requires adding a row and column to the

R factor; whereas the case $k+1 \geq m$ requires first deleting a column (or two) of $\hat{\Psi}_k$ before adding the newest limited-memory pair. In both cases, the columns of Q need not be formed nor stored. However, when $\hat{\Psi}_k$ is not full rank, the QR decomposition must be computed from scratch.

We begin by discussing the process to compute $\hat{\Psi}_{k+1}$ from $\hat{\Psi}_k$ when a new limited-memory pair is added to S and Y. The discussion considers the Broyden convex class of updates; however, comments are included at the end of each subsection regarding the SR1 case.

4.2.1. Adding a column to S and Y. Suppose $\hat{\Psi}_k = QR \in \Re^{n\times l}$ is full rank and we have stored $k+1$ limited-memory Broyden convex class updates such that $k+1 < m$, where m is the maximum number of limited-memory updates allowed to be stored by the limited-memory quasi-Newton method. Further, suppose we have computed a $(k+2)$nd pair (s_{k+1}, y_{k+1}). To update the QR decomposition, we augment $\hat{\Psi}_k$ with the two columns $\begin{pmatrix} B_0 s_{k+1} & y_{k+1}\end{pmatrix}$. This can be accomplished by using the procedure proposed by Gill et al. [14] for updating the QR factorization after a column is added. This method relies upon $\hat{\Psi}_k$ having full column rank. For completeness, this procedure is presented below in the context of adding two columns to $\hat{\Psi}_k$. As in the previous section, we assume that B_k is updated using the Broyden convex set of updates.

We begin by first adding the column $B_0 s_{k+1}$ to $\hat{\Psi}_k$; the same process may be followed to add the new last column, y_{k+1}. Suppose

$$\hat{\Psi}_k = Q \begin{pmatrix} R_1 \\ 0 \end{pmatrix}, \tag{4.3}$$

where $R_1 \in \Re^{l\times l}$. Moreover, suppose we insert $B_0 s_{k+1}$ into the final column of $\hat{\Psi}_k$ to obtain $\hat{\hat{\Psi}}_k$. Setting

$$\hat{\hat{\Psi}}_k = Q \begin{pmatrix} R_1 & u_1 \\ 0 & u_2 \end{pmatrix}$$

yields that

$$B_0 s_{k+1} = Qu \quad \text{with} \quad u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}, \tag{4.4}$$

where $u_1 \in \Re^l$ and $u_2 \in \Re^{n-l}$. We now construct an orthogonal matrix H_1 such that

$$H_1 \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} u_1 \\ \eta \\ 0 \end{pmatrix}, \tag{4.5}$$

where $\eta = \pm\|u_2\|$, i.e.,

$$H_1 = \begin{pmatrix} I & 0 \\ 0 & \hat{H}_1 \end{pmatrix}$$

where $\hat{H}_1$ is a Householder matrix such that $\hat{H}_1 u_2 = \begin{pmatrix} \eta & 0\end{pmatrix}^T$. This choice of H_1 preserves the structure of R_1, i.e.,

$$H_1 \begin{pmatrix} R_1 & u_1 \\ 0 & u_2 \end{pmatrix} = \begin{pmatrix} R_1 & u_1 \\ 0 & \eta \\ 0 & 0 \end{pmatrix}.$$

Then, $\hat{\hat{\Psi}}_k = \hat{Q}\hat{R}$ is the QR decomposition of $\hat{\hat{\Psi}}_k$, where

$$\hat{R} = \begin{pmatrix} R_1 & u_1 \\ 0 & \eta \\ 0 & 0 \end{pmatrix},$$

and $\hat{Q} = QH_1^T$.

In this procedure, the matrices Q, $\hat{Q}$, and H_1 are not stored; moreover, the unknowns u_1 and η can be computed without explicitly using these matrices. Specifically, the relation in (4.4) implies $\hat{\Psi}_k^T B_0 s_{k+1} = \begin{pmatrix} R_1^T & 0 \end{pmatrix} Q^T Q u$, i.e.,

$$\hat{\Psi}_k^T B_0 s_{k+1} = R_1^T u_1. \tag{4.6}$$

Equation (4.6) is a square $l \times l$ system that can be solved for u_1 provided $\hat{\Psi}_k$ is full rank. Finally, the scalar η can be computed from the following relation obtained from combining equations (4.4) and (4.5): $\|B_0 s_{k+1}\|^2 = \| \begin{pmatrix} u_1 & \eta \end{pmatrix} \|^2$. This yields that $\eta^2 = \|B_0 s_{k+1}\|^2 - \|u_1\|^2$. This procedure can be repeated to add y_{k+1} to the new last column of $\hat{\hat{\Psi}}_k$, thereby updating the QR factorization of $\hat{\Psi}_k$ to $\hat{\Psi}_{k+1}$ with a new pair of updates $(B_0 s_{k+1}, y_{k+1})$.

The process of adding a new SR1 update to $\hat{\Psi}_k$ is simpler since $\hat{\Psi}_k$ is augmented by only one column: $y_k - B_0 s_k$.

4.2.2. The full-rank assumption. The process described above requires $\hat{\Psi}_k$ to be full rank so that there is a (unique) solution to (4.6). When $\hat{\Psi}_k$ is not full rank, the QR decomposition must be computed from scratch. Fortunately, there is an *a priori* way to determine when there is no unique solution: The matrix $\hat{\hat{\Psi}}_k$ has full rank if and only if R_1 in (4.6) is invertible; in particular, the diagonal of R_1 is nonzero. When R_1 is singular, the process described above to update the QR decomposition for $\hat{\Psi}_{k+1}$ is skipped and the QR decomposition of $\hat{\Psi}_{k+1}$ should be computed from scratch at a cost of $2l^2(n - l/3)$ flops. The process described in Section 4.2.1 can be reinstated to update the QR decomposition when the R_1 factor has nonzero diagonal entries, which may occur again once the limited-memory updates exceed the maximum number allowed, (i.e., $k \geq m$), and we are forced to delete the oldest pairs.

Similarly, when $\hat{\Psi}_k$ is ill-conditioned, R_1 will also be ill-conditioned with at least one relatively small diagonal entry. In this case, (4.6) should not be solved; instead, the QR factorization should be computed from scratch. As with the rank-deficient case, it is possible to know this *a priori*.

4.2.3. Deleting and adding columns to S and Y. In this section, we detail the process to update the QR factorization in an efficient manner when $\hat{\Psi}_k$ is full rank and $k + 1 \geq m$. As in the previous section, we assume we are working with the Broyden convex class of updates.

Suppose $\hat{\Psi}_k = QR$ and we have stored the maximum number $(k + 1)$ limited-memory pairs $\{(s_i, y_i)\}$, $i = 0, \ldots, k$ allowed by the limited-memory quasi-Newton method. Further, suppose we have computed a $(k + 2)$nd pair (s_{k+1}, y_{k+1}). The process to obtain an updated QR factorization of $\hat{\Psi}_{k+1}$ from $\hat{\Psi}_k$ can be viewed as a two step process:

1. Delete a column of S and Y.
2. Add a new column to S and Y.

For the first step, we use ideas based on Daniel et al. [9] and Gill et al. [14]. Consider the Broyden class of updates. Suppose we rewrite $\hat{\Psi}_k$ and R as

$$\hat{\Psi}_k = \begin{pmatrix} B_0 s_0 & y_0 & \tilde{\Psi}_k \end{pmatrix} \quad \text{and} \quad R = \begin{pmatrix} r_1 & r_2 & \tilde{R} \end{pmatrix}, \tag{4.7}$$

where $\tilde{\Psi}_k \in \Re^{n\times(l-2)}$ and $\tilde{R} \in \Re^{n\times(l-2)}$. This gives that

$$\hat{\Psi}_k = \begin{pmatrix} B_0 s_0 & y_0 & \tilde{\Psi}_k \end{pmatrix} = Q \begin{pmatrix} r_1 & r_2 & \tilde{R} \end{pmatrix}.$$

Deleting the first two columns of $\hat{\Psi}_k$ yields the matrix $\bar{\Psi}_k = Q\tilde{R}$. Notice that $\tilde{R}$ has zeros beneath the second subdiagonal. For clarity, we illustrate the nonzero entries of $\tilde{R}$ for the case $n = 8$ and $k = 2$:

$$\begin{pmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \tag{4.8}$$

where * denotes possible nonzero entries. Givens rotations can be used to zero out the entries beneath the main diagonal in $\tilde{R}$ at a cost of $24k^2 - 36k$. (In the above example, eight entries must be zeroed out to reduce (4.8) to upper triangular form; more generally, $4(k-1)$ entries must zeroed out to reduce $\tilde{R}$ to upper triangular form.) Let $G_{i,j} \in \Re^{n\times n}$ denote the Givens matrix that zeros out the (i,j)th component of $\tilde{R}$, and suppose $\hat{G}$ is given by

$$\hat{G} \triangleq G_{2j-1,2j-2} G_{2j,2j-2} \cdots G_{2,1} G_{3,1}.$$

Then, $\hat{R} \triangleq \hat{G}\tilde{R}$ is an upper triangular matrix. This yields the QR decomposition of the matrix $\bar{\bar{\Psi}}_k$ defined as follows:

$$\bar{\bar{\Psi}}_k \triangleq \hat{Q}\hat{R}, \tag{4.9}$$

where $\hat{Q} = Q\hat{G}^T \in \Re^{n\times n}$ is orthogonal and $\hat{R} \in \Re^{n\times(l-2)}$ is an upper triangular matrix. With the computation of $\bar{\bar{\Psi}}_k$ we have completed the first step. Notice that neither Q nor $\hat{Q}$ must be stored in order to obtain $\hat{R}$.

For the second step, the QR factorization of $\hat{\Psi}_{k+1}$ can be obtained from $\bar{\bar{\Psi}}_k$ using the procedure outlined in Section 4.2.1.

The process required for SR1 updates is simpler than for the Broyden convex class of updates since it is only a rank-one update. That is, only one column of $\hat{\Psi}_k$ must be deleted to remove the old pair of updates and only one column must be added to incorporate the newest pair of updates.

5. Numerical experiments. In this section, we demonstrate the accuracy of the proposed method implemented in MATLAB to compute the eigenvalues of limited-memory quasi-Newton matrices. For the experiments, we considered limited-memory SR1 matrices and three limited-memory members of the Broyden convex class of updates; namely, limited-memory BFGS updates ($\phi = 0$), limited-memory DFP updates

($\phi = 1$), and limited-memory updates obtained by selecting $\phi = 0.5$ in (3.1). The number of limited-memory updates was set to 5 and $\gamma = 3$. The pairs S and Y were generated using random data. In addition, in the case of the Broyden convex class of updates, the updates were ensured to generate a positive definite matrix by redefining s_i as follows: $s_i = \text{sign}(s_i^T y_i) s_i$ whenever $s_i^T y_i < 0$ for each $i \in \{1, \ldots, 5\}$.

We report the results of following three numerical experiments on each quasi-Newton matrix:

Experiment 1. The eigenvalues of the quasi-Newton matrix were computed using the procedure outlined in Section 4.1.
Experiment 2. A new quasi-Newton pair was generated, adding a column to both S and Y. The procedure outlined in Section 4.2.1 was used to update Ψ, and then the eigenvalues were recomputed using the procedure outlined in Section 4.1.
Experiment 3. The first columns of S and Y were deleted, simulating the case when the oldest pair of updates is discarded. Then, a new quasi-Newton pair was generated, adding a new column to the end of both S and Y. The procedure outlined in Section 4.2.3 was used to update Ψ, and then the eigenvalues were recomputed using the procedure outlined in Section 4.1.

To determine the accuracy of the proposed method, we explicitly formed each quasi-Newton matrix and used the MATLAB `eig` command to compute its actual eigenvalues. Due to memory limitations in computing actual eigenvalues to test the proposed method, we restricted the matrix sizes to $n \leq 5000$. For each of the three experiments, we report the size of the quasi-Newton matrix ("n") and the relative error in the computed eigenvalues measured by the infinity norm ("RE Experiment 1", "RE Experiment 2", and "RE Experiment 3"); that is, for each experiment the relative error was computed as

$$\text{RE} = \frac{\|(D + \gamma I) - \Lambda\|_\infty}{\|\Lambda\|_\infty},$$

where $(D + \gamma I)$ is as in (4.2) and Λ is the matrix of eigenvalues obtained using the MATLAB `eig` command.

In Table 1, we report the results when B was taken to be randomly-generated limited-memory SR1 matrices of sizes $n = 100, 500, 1000$ and 5000. For each matrix, the relative error in computing the eigenvalues using the proposed method is very small (column 1). As seen in column 2, the relative error remained very small after a new column was added to both S and Y, Ψ was updated using using Section 4.2.1, and the eigenvalues were recomputed using the procedure outlined in Section 4.1. Finally, column 3 shows that the relative error remained very small after discarding the first stored quasi-Newton pair and adding a new column to both S and Y using the procedures outlined in Section 4.2.3 and Section 4.1.

TABLE 1
Summary of results when B is a limited-memory SR1 matrix.

n	RE Experiment 1	RE Experiment 2	RE Experiment 3
100	1.92439e-15	2.07242e-15	2.81256e-15
500	4.88498e-15	4.44089e-15	6.21725e-15
1000	8.14164e-15	7.99361e-15	7.84558e-15
5000	1.71714e-14	1.98360e-14	1.68754e-14

Table 2 reports the results when B was a randomly-generated limited-memory BFGS matrix of sizes $n = 100, 500, 1000$, and 5000. In all cases, the proposed method computed the eigenvalues to high accuracy. Table 3 reports the results when B was a randomly-generated limited-memory DFP matrix of various sizes. As in Tables 1 and 2, the proposed method computed the eigenvalues of these matrices to high accuracy in each experiment.

TABLE 2
Summary of results when B is a limited-memory BFGS matrix.

n	RE Experiment 1	RE Experiment 2	RE Experiment 3
100	5.53332e-16	1.21039e-16	7.86896e-16
500	6.35220e-16	4.28038e-16	5.86555e-16
1000	1.13708e-15	2.39590e-15	1.62325e-15
5000	1.14773e-15	3.39882e-15	1.30101e-15

TABLE 3
Summary of results when B is a limited-memory DFP matrix.

n	RE Experiment 1	RE Experiment 2	RE Experiment 3
100	1.69275e-15	2.05758e-16	3.65114e-16
500	9.58309e-16	6.19241e-16	2.10460e-15
1000	4.15522e-15	1.30844e-14	1.72417e-14
5000	2.27937e-15	1.20206e-14	2.97026e-15

Finally, Table 4 reports the results when B obtained using the Broyden convex class of updates with $\phi = 0.5$. In all experiments with this type of update, the proposed method was able to compute all the eigenvalues to high accuracy.

TABLE 4
Summary of results when B is a limited-memory member of the Broyden class of convex updates with $\phi = 0.5$.

n	RE Experiment 1	RE Experiment 2	RE Experiment 3
100	5.11757e-15	9.05737e-15	6.02940e-16
500	1.11222e-15	4.90513e-15	1.60814e-15
1000	1.76830e-15	2.83112e-15	2.18559e-15
5000	9.86622e-15	2.95003e-15	5.88569e-15

6. Concluding remarks. In this paper we produced the compact formulation of quasi-Newton matrices generated by the Broyden convex class of updates. Together with the QR factorization, this compact representation was used to compute the eigenvalues of any member of this class of updates. In addition, we presented an efficient procedure to update the QR factorization when a new pair of updates for the quasi-Newton matrix is computed. With this approach we are able to substantially reduce the computational costs of computing the eigenvalues of quasi-Newton matrices. Applications of this work are the subject of current research. Code and drivers used for this paper can be found on the following website:

`http://users.wfu.edu/erwayjb/software.html`.

Appendix A. In [1, 2], Apostolopoulou et al. find explicit formulas for computing the eigenvalues of a BFGS matrix when **at most** two limited-memory quasi-Newton pairs are used to update an initial B_0. While the methods in this paper are not limited to two updates and can be applied to SR1 matrices and any matrix generated using the Broyden convex class of updates, we show that the results found in [1, 2] for the case of one update can be derived using the technique proposed in this paper.

Without loss of generality, Apostolopoulou et al. derive a formula for computing the eigenvalues of the following matrix obtained after applying one update:

$$B_1 = \frac{1}{\theta_0} I - \frac{1}{\theta_0} \frac{s_0 s_0^T}{s_0^T s_0} + \frac{y_0 y_0^T}{s_0^T y_0}, \quad \text{where} \quad \theta_0 = \frac{s_0^T s_0}{s_0^T y_0}. \tag{A.1}$$

The compact formulation of B_1 is given by $B_1 = B_0 + \Psi_0 M_0 \Psi_0^T$ where

$$\Psi_1 = \begin{bmatrix} B_0 s_0 & y_0 \end{bmatrix} \quad \text{and} \quad M_0 = \begin{bmatrix} -s_0^T B_0 s_0 & 0 \\ 0 & s_0^T y_0 \end{bmatrix}^{-1}.$$

For notational simplicity, we drop the subscript $k = 0$ for the duration of Appendix A. We now compute the eigenvalues of $B_1 = (1/\theta) I + \Psi M \Psi^T$ using the QR factorization of Ψ and show that these eigenvalues are the same as those obtained in [1, 2] by comparing the characteristic polynomials.

The QR factorization can be computed using Householder transformations. The first Householder transformation zeros out all the elements in the first column of Ψ below the first entry. Let $v_1 \triangleq \Psi e_1 - \|\Psi e_1\| e_1$, where e_1 denotes the first canonical basis vector. Since $\Psi e_1 = (1/\theta) s$, then

$$Q_1 = I - \frac{2}{v_1^T v_1} v_1 v_1^T$$

is such that

$$Q_1 (\Psi e_1) = Q_1 \left(\frac{1}{\theta} \right) s = \frac{\|s\|}{\theta} e_1.$$

(For more details on constructing Householder matrices, see e.g., [16].)

Using the definition of Q_1 and v_1 gives that

$$Q_1 \Psi = Q_1 [(1/\theta) s \quad y] = \left[\tfrac{\|s\|}{\theta} e_1 \quad y - \tfrac{2 v_1^T y}{v_1^T v_1} v_1 \right] = \left[\begin{array}{c|c} \frac{\|s\|}{\theta} & \\ 0 & \\ \vdots & y - \frac{2 v_1^T y}{v_1^T v_1} v_1 \\ 0 & \end{array}\right]. \tag{A.2}$$

The second Householder transformation zeros out all entries in the second column below the second row. Let

$$u_2 = \begin{bmatrix} (y - \frac{2 v_1^T y}{v_1^T v_1} v_1)^T e_2 \\ \vdots \\ (y - \frac{2 v_1^T y}{v_1^T v_1} v_1)^T e_n \end{bmatrix}.$$

Defining $v_2 \triangleq u_2 - \|u_2\|e_1$, then

$$\hat{Q}_2 = I - \frac{2}{v_2^T v_2} v_2 v_2^T$$

is such that $\hat{Q}_2 u_2 = \|u_2\|e_1$, and

$$Q_2 \triangleq \begin{bmatrix} 1 & 0 \\ 0 & \hat{Q}_2 \end{bmatrix}$$

is such that

$$R \triangleq Q_2 Q_1 \Psi = \begin{bmatrix} \frac{\|s\|}{\theta} & \left(y - \frac{2v_1^T y}{v_1^T v_1} v_1\right)^T e_1 \\ 0 & \|u_2\| \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix} \tag{A.3}$$

is an upper triangular matrix. If $Q \triangleq Q_1^T Q_2^T$ then by (A.3)

$$B_1 = Q\left(\frac{1}{\theta} I + RMR^T\right) Q^T. \tag{A.4}$$

Let $R_1 \in \Re^{2\times 2}$ be the first 2×2 block of R, i.e.,

$$R_1 = \begin{bmatrix} \frac{\|s\|}{\theta} & \left(y - \frac{2v_1^T y}{v_1^T v_1} v_1\right)^T e_1 \\ 0 & \|u_2\| \end{bmatrix}.$$

The matrix R_1 can be further simplified by noting that

$$\begin{aligned}
\left(y - \frac{2v_1^T y}{v_1^T v_1} v_1\right)^T e_1 &= \frac{y^T e_1 v_1^T v_1}{v_1^T v_1} - \frac{2v_1^T y}{v_1^T v_1} v_1^T e_1 \\
&= \frac{y^T e_1 (s - \|s\|e_1)^T (s - \|s\|e_1) - 2(s - \|s\|e_1)^T y (s - \|s\|e_1)^T e_1}{(s - \|s\|e_1)^T (s - \|s\|e_1)} \\
&= \frac{2y^T e_1 \|s\|(\|s\| - s^T e_1) + 2(s - \|s\|e_1)^T y(\|s\| - s^T e_1)}{2\|s\|(\|s\| - s^T e_1)} \\
&= \frac{y^T e_1 \|s\| + (s - \|s\|e_1)^T y}{\|s\|} \\
&= \frac{s^T y}{\|s\|},
\end{aligned}$$

and thus,

$$R_1 = \begin{bmatrix} \frac{\|s\|}{\theta} & \frac{s^T y}{\|s\|} \cdot \\ 0 & \|u_2\| \end{bmatrix}.$$

Rewriting (A.4) using R_1 yields

$$B_1 = Q \begin{bmatrix} \frac{1}{\theta} I_2 + R_1 M R_1^T & 0 \\ 0 & \frac{1}{\theta} I_{n-2} \end{bmatrix} Q^T,$$

implying that the eigenvalues of B_1 are the union of eigenvalues of the two diagonal blocks in B_1. Note that the leading 2×2 block can be simplified as follows:

$$\begin{aligned}\frac{1}{\theta}I_2 + R_1 M R_1^T &= \frac{1}{\theta}I - \begin{bmatrix} \frac{\|s\|}{\theta} & \frac{s^T y}{\|s\|} \\ 0 & \|u_2\| \end{bmatrix} \begin{bmatrix} \frac{\theta}{s^T s} & 0 \\ 0 & -\frac{1}{s^T y} \end{bmatrix} \begin{bmatrix} \frac{\|s\|}{\theta} & 0 \\ \frac{s^T y}{\|s\|} & \|u_2\| \end{bmatrix} \\ &= \frac{1}{\theta}I - \begin{bmatrix} \frac{1}{\theta} - \frac{(s^T y)}{\|s\|^2} & -\frac{\|u_2\|}{\|s\|} \\ -\frac{\|u_2\|}{\|s\|} & -\frac{\|u_2\|^2}{s^T y} \end{bmatrix} \\ &= \begin{bmatrix} \frac{(s^T y)}{\|s\|^2} & \frac{\|u_2\|}{\|s\|} \\ \frac{\|u_2\|}{\|s\|} & \frac{1}{\theta} + \frac{\|u_2\|^2}{s^T y} \end{bmatrix}.\end{aligned}$$

The characteristic polynomial of leading 2×2 block of B_1 is given by

$$\det\left(\frac{1}{\theta}I + R_1 M R_1^T - \lambda I\right) = \lambda^2 - \lambda\left(\frac{1}{\theta} + \frac{\|u_2\|^2}{s^T y} + \frac{s^T y}{\|s\|^2}\right) + \frac{1}{\theta}\frac{s^T y}{\|s\|^2}. \tag{A.5}$$

Finally, since

$$\left[\left(y - \frac{2v_1^T y}{v_1^T v_1}v_1\right)^T e_1\right]^2 + \|u_2\|^2 = \|Q_2 Q_1 y\|^2 = y^T y \quad \text{and} \quad \left[\left(y - \frac{2v_1^T y}{v_1^T v_1}v_1\right)^T e_1\right]^2 = \frac{(s^T y)^2}{\|s\|^2},$$

then (A.5) simplifies to

$$\lambda^2 - \frac{\lambda}{\theta}\left(1 + \theta\frac{y^T y}{s^T y} + \frac{1}{\theta^2}\right). \tag{A.6}$$

Thus, the characteristic polynomial of B_1 is given by

$$p(\lambda) = \left(\lambda^2 - \frac{\lambda}{\theta}\left(1 + \theta\frac{y^T y}{s^T y}\right) + \frac{1}{\theta^2}\right)\left(\lambda - \frac{1}{\theta}\right)^{n-2},$$

which is the same as the characteristic polynomial derived in [1, Equation 4] and [2, Equation 9].

REFERENCES

[1] M. S. Apostolopoulou, D. G. Sotiropoulos, C. A. Botsaris, and Panayiotis E. Pintelas, *A practical method for solving large-scale TRS*, Optimization Letters, 5 (2011), pp. 207–227.

[2] M. S. Apostolopoulou, D. G. Sotiropoulos, and P. Pintelas, *Solving the quadratic trust-region subproblem in a low-memory BFGS framework*, Optimization Methods Software, 23 (2008), pp. 651–674.

[3] J. R. Bunch, C. P. Nielsen, and D. C. Sorensen, *Rank-one modification of the symmetric eigenproblem*, Numerische Mathematik, 31 (1978), pp. 31–48.

[4] O. Burdakov, L. Gong, Y.-X. Yuan, and S. Zikrin, *On efficiently combining limited memory and trust-region techniques*, Tech. Report 2013:13, Linkping University, Optimization, 2013.

[5] J. V. Burke, A. Wiegmann, and L. Xu, *Limited memory BFGS updating in a trust-region framework*, technical report, University of Washington, 1996.

[6] R. H. Byrd, J. Nocedal, and R. B. Schnabel, *Representations of quasi-Newton matrices and their use in limited-memory methods*, Math. Program., 63 (1994), pp. 129–156.

[7] A. R. Conn, N. I. M. Gould, and Ph. L. Toint, *Convergence of quasi-newton matrices generated by the symmetric rank one update*, Math. Program., 50 (1991), pp. 177–195.

[8] A. R. Conn, N. I. M. Gould, and Ph. L. Toint, *Trust-Region Methods*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000.
[9] J. W. Daniel, W. B. Gragg, L. Kaufman, and G. W. Stewart, *Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization*, Math. Comput., 30 (1976), pp. 772–795.
[10] L. C. W. Dixon, *Quasi-Newton algorithms generate identical points*, Math. Program., 2 (1972), pp. 383–387.
[11] J. B. Erway, V. Jain, and R. F. Marcia, *Shifted limited-memory DFP systems*, in Signals, Systems and Computers, 2013 Asilomar Conference on, Nov 2013, pp. 1033–1037.
[12] R. Fletcher, *Practical Methods of Optimization*, Wiley-Interscience [John Wiley & Sons], New York, 2001.
[13] D. M. Gay, *Computing optimal locally constrained steps*, SIAM J. Sci. Statist. Comput., 2 (1981), pp. 186–197.
[14] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders, *Methods for modifying matrix factorizations*, Math. Comput., 28 (1974), pp. 505–535.
[15] G. H. Golub, *Some modified matrix eigenvalue problems*, SIAM Rev., 15 (1973), pp. 318–334.
[16] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, Maryland, third ed., 1996.
[17] I. Griva, S. G. Nash, and A. Sofer, *Linear and nonlinear programming*, Society for Industrial and Applied Mathematics, Philadelphia, 2009.
[18] N. J. Higham, *Accuracy and stability of numerical algorithms*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2002.
[19] T.G. Kolda, D.P. O'leary, and L. Nazareth, *BFGS with update skipping and varying memory*, SIAM Journal on Optimization, 8 (1998), pp. 1060–1083.
[20] D. Kressner, *Numerical methods for general and structured eigenvalue problems*, Lecture notes in computational science and engineering, Springer, Berlin, Heidelberg, 2005.
[21] D.C. Liu and J. Nocedal, *On the limited memory BFGS method for large scale optimization*, Mathematical programming, 45 (1989), pp. 503–528.
[22] D.G. Luenberger and Y. Ye, *Linear and nonlinear programming*, vol. 116, Springer, 2008.
[23] J. J. Moré and D. C. Sorensen, *Computing a trust region step*, SIAM J. Sci. and Statist. Comput., 4 (1983), pp. 553–572.
[24] S.G. Nash and J. Nocedal, *A numerical study of the limited memory BFGS method and the truncated-newton method for large scale optimization*, SIAM Journal on Optimization, 1 (1991), pp. 358–372.
[25] J. Nocedal, *Updating quasi-newton matrices with limited storage*, Mathematics of computation, 35 (1980), pp. 773–782.
[26] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer-Verlag, New York, 1999.
[27] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.